REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* September 2004	**2. REPORT TYPE** Technical Paper	**3. DATES COVERED** *(From - To)* 3 May 2004 -
4. TITLE AND SUBTITLE A Memory Lattice of Quadrature Memory Filter Groups		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Laird, Daniel T. 412TW/ENTI (ARTM)		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES) 412TW/ENTI Air Force Flight Test Center (AFFTC) Edwards AFB, CA 93524		**8. PERFORMING ORGANIZATION REPORT NUMBER** PA-04100
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) 412TW/ENTI Air Force Flight Test Center (AFFTC) Edwards AFB, CA 93524		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S) N/A

12. DISTRIBUTION / AVAILABILITY STATEMENT
A Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
CC: 012100 CA: Air Force Flight Test Center Edwards AFB

14. ABSTRACT
This paper discusses a memory architecture that realizes a group structure of data filters on a lattice of control strings (sets). The lattice filters concatenate & parse control strings for memory I/O using a method that in various ways 'mirrors' orthogonal quadrature signal filtering from wavelet theory (thus the name). The lattice indicates algebraic structures, of partial orders & set inclusion, that filters perform on the control string 'sets'. There are many ways to realize memorization of controls for system configuration: this is one way that I found interesting for its algebraic structure.

15. SUBJECT TERMS
Data filters, Concatenate-parse; Quadrature (orthogonal)

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Daniel T. Laird
a. REPORT UNCLASSIFIED	**b. ABSTRACT** UNCLASSIFIED	**c. THIS PAGE** UNCLASSIFIED	Unclassified Unlimited	27	**19b. TELEPHONE NUMBER** *(include area code)* 661-277-1610

A Memory Lattice
of
Quadrature Filter Groups

Abstract

Automated system configuration requires a memory, which stores, restores and recollects control parameters necessary for the activation of hardware. Such memory structures are usually designed as Finite State Machines (FSM) or State Mechanisms. Mechanistic memory is perhaps different in kind from that of an organism; in this paper we shall investigate a model of memory influenced by organic structures, i.e., DNA-RNA cellular dynamics.

Although influenced by organic structure, this memory architecture is a hierarchy of mechanistic structures called 3-level[i] languages applicable to mechanistic control, and was in fact realized as a memorization scheme for an RF Test Executive (RTEx[ii]) as the control of Doppler field generation. This control-memorization scheme is applicable for any automatic control; it is also part of the Airborne Icing Tanker control & environmental adaptation scheme.

Introduction

In this paper we discuss some investigations into memory structures; in particular we reexamine the idea of memorization as the integration of newer memories over older, active memories as content modifiers of fixed context. We may view this as memory reactivation and reintegration given new input stimuli. Also, the memory significance, or 'weight' as a specific energy measure on the data is an element of a memory lattice.

To make the investigation specific we review an existing memory structure of RTEx that coordinates testing of semi-active seekers in simulated dynamic RF environments created by a network of signal generator-modulators and measured by a network of acquisition units. We investigate only the memorization dynamics for the generator-modulator control structures and leave the acquisition for another paper.

By making minor changes to the memorization scheme we shall attempt to examine 'continuous' memorization of control states during system (hardware) activation. In this approach we view memorization as a composite of data set **activity**: *construction*, *recollection* and *restoration*, for which construction & recollection, or what I designate as deconstruction, form a filter group. The inverse group action has a local energy measure which is an element of a global memory lattice.

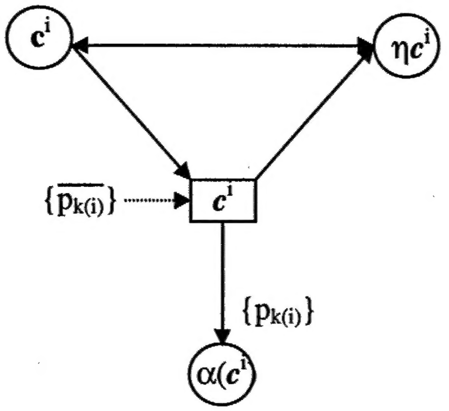
RTEx originally *configured*, *activated*, *stored* and *recalled* system states on operator command. For the memorization discussed below we eliminate the need for operator command and assume the system is programmed to memorize each time a control parameter changes by some set amount regardless of the source of change; this type of memorization is *learning on change*. RTEx recalls control structures coded as **string** memories: strings code clusters, i.e., a composite data structure of multiple types which code as a single string data type. The structure of a string memory is hierarchical, with the highest level string designated as $\mathbf{M}^n$; where n designates a 'name'. The memory name associates a memory, $\mathbf{M}^n$ with a Test Sequences: $S(\mathbf{M}^n)$. The sequence, $S(\mathbf{M}^n)$, code for hardware activation sequences of network of signal generators.

RTEx constructs, recollects, restores, modifies and controls $S(\mathbf{M})$ via an intuitive operator graphical user interface (GUI). Recollected & active memory strings are stored in a hierarchical construct of **modules**; each module has a set of control **clusters**, which are in turn composites of numeric & logical data types. The elements of a cluster form **parameter sets**. The original data parameter sets are input by the operator via the RTEx GUI and these may be stored, recalled & manipulated as required.

The highest level in the hierarchy of local control is the module; module configurations are stored as module, or **m**–strings. These **m**–strings are composites of cluster control string, or **c**-strings. As a module has various control clusters an store as 2nd order memories. The **c**–strings form 1st order memory structures over parameter sets, $\{p_k\}$, of elements coded as p–string units. The parameter sets are coded as 0th order memories. The energy measure is defined as the 'norm' of clusters, i.e., as the root of the sum of squared parameters: $\sqrt{\Sigma p_k}$. There are other energy measures one could use, e.g., the sum of the parameters. The total memory structure is a hierarchy of module memories coded as a 3rd order memory: $\mathbf{M}_{(j(i(k))}$. The j-index sequences modules; the i-index sequences clusters; & the k-index sequences parameters. To associate this structure with a name we designate the global memory as $\mathbf{M}^n{}_{(j(i(k))}$. For simplicity I will denote this as $\mathbf{M}^n{}_{jik}$, or simply suppress the index sets where they are understood.

Each $\mathbf{M}^n$ encapsulates several modules and each module encapsulates several clusters and each cluster encapsulates a set of control parameters. Thus, the memory architecture is a composite of parameter p–strings, $p(s)_{k(i)}$, cluster **c**–strings, $\mathbf{c}^{i(j)}$, and module **m**–strings, $\vec{\mathbf{m}}^j$; all of which are packaged as *local* strings and stored as a single *global* string, $\mathbf{M}^n$, in a Test Sequence initialization file: $f(\mathbf{M}^n{}_{jik})$. As the parameter index is specific to the ith cluster, the parameter set index, k, is a function of the ith cluster; the index is designated k(i); & similarly for i(j) indexing module clusters. The content of the file is the global **M**–string which codes all the configuration information of a Test Sequence $S(\mathbf{M}^n{}_{jik})$; the actual sequencing structure, or 'timing' is built into the RTEx. For convenience we will denote a test sequence simply as $S(\mathbf{M}^n)$ and a file as $f(\mathbf{M}^n)$, where the indexing is understood.

What we shall examine below is a data filter structure of the memorization architecture called **quadrature filters**. The filters are not part of the sequencing structure, they are sub-modules that communicate data between control clusters and initialization files – they are 'tangent' or 'conjugate' to the strings they process. These filters are distributed across the control modules as data **constructor** and **deconstructor**. The filters service a *core*, called the **activator**, with parameter sets; a module schematic, which we shall expound upon, is shown below.


$\mathbf{c}^{i}$
$\eta \boldsymbol{c}^{i}$
$\{\overline{p_{k(i)}}\}$
$\boldsymbol{c}^{i}$
$\{p_{k(i)}\}$
$\alpha(\boldsymbol{c}^{i}$

Note that the clusters serve a unique position in the module architecture as both source and sink of memory and activation parameter sets.

Control Modules

Each RTEx control module has three basic functions: hardware *activation*, control memory *construction* and memory *destruction* to control. These three functions are coded in the three sub-modules mentioned above. The RTEx interfaces with an operator via a module that contains all three sub-modules. The operator communicates with the system via control clusters, $\boldsymbol{c}^{i}$, as parameter acceptor.

The deconstructor is a composite structure that opens string memories and reduces sub-strings to the next lower order memory. This data filter we shall naturally designate an *extractor-reducer*. Thus the deconstructor 'unpackages' module memories into the next lower order memory; the operation is denoted:

$$\psi_s[\vec{\boldsymbol{v}}^{sr}] \Rightarrow \cup^{rs}\mathbf{u}^{s};$$

where **u** and $\vec{\boldsymbol{v}}^{sr}$ designates general string memories whose structures are explicated below.

The constructor is a composite structure that extends sub-strings into the next higher order memory, then closes the memory at the completion of the extension; thus, we designate this operation the *extender-contractor*. The contractor 'packages' string memories for storage in higher order memory; the contractor operation is denoted:

$$\psi^r[\cup^{rs}\mathbf{u}^s] \Rightarrow \vec{\boldsymbol{v}}^{sr}.$$

The *deconstructor-constructor* action is that of reciprocal data filters for the activator control I/O.

The *activation* of the module control is via a separate sub-module that operates on the recollected memory and may also accept operator input; thus the activator serves as memory source and sink, as well as a sink for 'external input'. We discuss all in detail below.

Constructor

Control clusters are the elements of modules; clusters are composites of parameters of various data types. Each cluster of parameters has a replica, $\eta \boldsymbol{c}^i(\{p_{k(i)}\})$ and this replica is mapped into a **c**–string. The 1st level of contraction maps the *parameter set*, $\{p_{k(i)}\}$, of delimited strings into a set of strings: $\{:p(s)_{k(i)}\}$; this set is concatenated as the contents of the **c**–string. Thus, the parameter set of a cluster, $\boldsymbol{c}^i$ is replicated in a cluster replica, $\eta \boldsymbol{c}^i$, transformed into strings and concatenated as a **c**–string: $\mathbf{c}^i$; this parameter extension is a two step process of extension:

$$\{p_{k(i)}\} \rightarrow \boldsymbol{c}^i(\{p_{k(i)}\}) \rightarrow \eta \boldsymbol{c}^i(\{p_{k(i)}\})$$

and translation:

$$\varphi^i[\eta \boldsymbol{c}^i(\{p_{k(i)}\})] \Rightarrow \mathbf{c}^i(p(s)_{k(i)});$$

where,

$\mathbf{c}^i \equiv p(s)^i_1|\ldots|p^i(s)_{k(i)};\ i,k(i)\in N.$

Here, $p(s)_{k(i)}$ designates the string representation of kth parameter in the ith cluster; k(i) is the cluster parameter set index of length $k = |k(i)|$; there is one such set for each cluster index i. The length, k, varies from cluster to cluster and is a small natural number $0\leq k\leq n$; n is denotes the maximum number of cluster parameters.

Note that the replica implicates two clusters over the parameter sets: the user control and its replica; the RTEx GUI has a composite of a cluster and its hidden replica; we denote this:

$\hat{\mathbf{c}} \equiv c(1,\eta).$

The replica is a duplicate of the control cluster hidden from the operator; the replica encapsulates the input parameters which are then translated into strings and concatenated for further processing. The mapping of cluster replica, ηc to a **c**–string memory is both injective and surjective, i.e., bijective:

$\mathbf{c}(p(s))\Leftrightarrow\eta c(p).$

The 2nd level of construction extends **c**–strings into local module string, or **m**–string: $\ddot{\mathbf{m}}^j$; the construction single layered and designated:

$\phi^j[\mathbf{c}^i] \Rightarrow \mathbf{c}^{ji}: \cup^{ji}\,\mathbf{c}^{ji}$

where,

$\ddot{\mathbf{m}}^j \equiv \mathbf{c}^{j0};\ldots;\mathbf{c}^{jn};\ j,n\in N.$

Here, $1\leq j\leq m$ indexes modules and $1\leq i\leq n$. The total local module construction is designated:

$\phi^j \circ \varphi^i[\{p_{k(i)}\}] = \phi^j[\mathbf{c}^i] \Rightarrow \mathbf{c}^{ji}: \cup^{ji}\,\mathbf{c}^{ji} \Rightarrow \ddot{\mathbf{m}}^j,$

where,

$\vec{\mathbf{m}}^j \equiv \mathbf{c}^{j0};\ldots;\mathbf{c}^{jn} = p(s)^{j0}{}_1|\ldots|p(s)^{j0}{}_{k(0)};\ldots;p(s)^{jn}{}_1|\ldots|p(s)^{j0}{}_{k(n)};$

$|p^{ji}{}_{k(i)}| \in \mathrm{R};\ j,i,k(i)\in \mathrm{N}.$

As each module configuration is part of a system state configuration comprised of several modules, there is a 3rd level construction of all the local **m**–strings which inserts them in a global string, $\mathbf{M}:\mathbf{M}\supset\vec{\mathbf{m}}^j$; ote that the global string is a hierarchy of inclusion: $\mathbf{M}\supset\vec{\mathbf{m}}^j\supset\mathbf{c}^i\supset p_{k(i)}$. The memory has the structure of an algebraic object called a lattice (cf. below).

The memory construction is a sequence of compositions: an organization of the internal elements of the 3rd order hierarchy. The algorithm is:

$$\gamma^{\mathbf{M}}\circ\phi^j\circ\varphi^i[\{p_{k(i)}\}]\Rightarrow\gamma^{\mathbf{M}}\circ\phi^j[\mathbf{c}^i]\Rightarrow\gamma^{\mathbf{M}}[;^{ji}\mathbf{c}^{ji}]\Rightarrow\gamma^{\mathbf{M}}[\vec{\mathbf{m}}^j]\Rightarrow:\vec{\mathbf{m}}^j\equiv\mathbf{M}^{ji}{}_{k(i)}.$$

We designate this composition as a simple juxtaposition:

$$\gamma^{\mathbf{M}}\phi^j\varphi^i[\{p_{k(i)}\}]\Rightarrow\mathbf{M}^{ji}{}_{k(i)},$$

when composition is understood. This is a 3rd order memory, comprised of 2nd order modules and 1st order controls, where the elements are 0th order parameters.

Each memory is assigned a basis dimension within the lattice. To index the global memory we designate this memory in terms of its parameters as

$$\mathbf{M}^{ji}{}_{k(i)}\equiv:\vec{\mathbf{m}}^j=:;\mathbf{c}^{ji}=:;p^{ji}{}_{k(i)}|\ i,j,k(i)\in\mathrm{N}.$$ [iii]

For simplicity we suppress delimiters and write address the global memory as

$$\mathbf{M}^{ji}{}_{k(i)}=\mathbf{M}(p^{ji}{}_{k(i)})\ i,j,k(i)\in\mathrm{N}.$$

This is the global memory **M**–string; this string contains all the control information for the system. This structure shows explicitly the memory as a 3rd order memory of parameters.

Deconstructor

The deconstructor works in opposition to the constructor, as it opens, extracts higher order memory and expands the contents for the lower order memory. Level 1 *opens* a file that contains global **M**–string, extracts **M** and expands *all* **m**–strings from the global **M**–string:

$$\gamma_M[\mathbf{M}^{ji}{}_{k(i)}] \Rightarrow \vec{\mathbf{m}}^j.$$

Each local **m**–string contains all control clusters for local modules; thus, level 2 expands $\vec{\mathbf{m}}^j$ into *all* **c**–strings:

$$\phi_j[\,\vec{\mathbf{m}}^j\,] = \phi_j[\mathbf{c}^{ji}] \Rightarrow \cup^{ji}\mathbf{c}^i.$$

This represents all $|i|$ control strings, $\mathbf{c}^i$, in the jth module. Level 3 expands all **c**–strings across all cluster and replicas. The elements of each **c**–string are expanded across the cluster replica, and the replica maps into the cluster; this is the alignment of control parameters with the respective cluster:

$$\cup^{ji}\varphi_i[\mathbf{c}^i(\mathrm{p(s)}_{k(i)})] \Rightarrow \cup^{ji}\eta\boldsymbol{c}^i(\{\mathrm{p}_{k(i)}\})$$

$$\eta\boldsymbol{c}^i(\{\mathrm{p}_{k(i)}\}) \rightarrow \boldsymbol{c}^i(\{\mathrm{p}_{k(i)}\}).$$

Since we need not expand across the replica and then map to the cluster, we can expand across both simultaneously; thus the recall can be shown as a composite distribution:

$$\varphi_i[\mathbf{c}^i(\mathrm{p(s)}_{k(i)})] \Rightarrow (1,\eta)\boldsymbol{c}^i(\{\mathrm{p}_{k(i)}\}).$$

In this way we have a 'complex cluster' – control and replica. The present structure maps **c**–strings to replicas. The total action is to produce control contents, which form the entire set union:

$$\mu{:}\{\mathrm{p}^{ji}{}_{k(i)}\} \equiv \cup^{ji}{}_{k(i)}|\mathrm{p}^{ji}{}_{k(i)}|.$$

This is called the ideal measure set of the generated Doppler field. Thus the end of destruction is the distribution of all parameters across the control clusters. In simplified notation we write the total filter action as:

$$\varphi_i \circ \phi_j \circ \gamma_M[\mathbf{M}^n_{jik}] = \varphi_i \circ \phi_j[\vec{\mathbf{m}}^j] = \varphi_i[\mathbf{c}^i] \Rightarrow \mu{:}\{p^{ji}_{k(i)}\}.$$

Substituting juxtaposition for composition:

$$\varphi_i\phi_j\gamma_M[\mathbf{M}^n_{jik}] \Rightarrow \mu{:}\{p^j_{Ik(i)}\}.$$

The entire memory structure expansion, or destruction represented as parameter measures is:

$$\mathbf{M}^n_{jik} \equiv \vec{\mathbf{m}}^0{:}\ldots{:}\vec{\mathbf{m}}^m = (\mathbf{c}^{00};\ldots;\mathbf{c}^{0p}){:}\ldots{:}(\mathbf{c}^{m0};\ldots;\mathbf{c}^{mq}),$$

$$\mathbf{M}^n_{jik} = (p^{00}{}_1|\ldots|p^{00}{}_{k(0)});\ldots;(p^{00}{}_1|\ldots|p^{00}{}_{k(p)})){:}\ldots$$

$$\ldots{:}((p^{m0}{}_1|\ldots|p^{m0}{}_{k(0)});\ldots;(p^{m0}{}_1|\ldots|p^{m0}{}_{k(q)})).$$

The control parameters (ideal measures) designate the desired field characteristics; thus the control parameters are extracted as the potential measures of the Doppler field. In this way the coded global **M**–string is *equivalent* to the control static structure, i.e., its total *measure set*: $\mu{:}\{p^{ji}_{k(i)}\}$.

The global string contains all the *measure* required for system configuration or state; but the $S(\mathbf{M}^n)$ structure also includes an *order* of execution. The order is contained in the module and control *activation sequencing*; we denote this total sequencing as

$$S(\mu) \equiv S^{ji}_{k(i)}(p^{ji}_{k(i)}))) = S^j \circ S^i \circ S_{k(i)}(p^{ji}_{k(i)}).$$

$S_{k(i)}(p^{ji}_{k(i)})$ is the k-sequence through the ith cluster in jth module. Note that there are many sequences ($S(\mathbf{M}^n)$) possible through the memory set $\mathbf{M}^n$, or its equivalent: the measure set, $S(\mu)$. The number of sequences is

$$\#(S(\mu)) = |i|!|j|!|k(i)|!$$

Of course, not all sequences make sense for the generation of Doppler fields. Those that do make sense are quite limited in number.

Activation

The activation of the jth module, $\mathbf{m}^j$, is denoted,

$$\alpha(\boldsymbol{m}^j) = \alpha(\boldsymbol{c}^{ji}) = \alpha(p^{j0}_{0},\dots p^{j0}_{r};\dots;p^{jq}_{0},\dots p^{jq}_{s}),$$

where $\boldsymbol{c}^{ji}$ is the ith cluster of the jth module and p^{j0}_{s} is the sth (k(0) = s) parameter on the 0th cluster of the jth module. We assume a natural sequence order. Also, we assume all are activated in some sequence on j, then i, then k(i). We assume that

$$\alpha(;\boldsymbol{c}^{ji}) = ;\alpha(\boldsymbol{c}^{ji}) = ;\alpha(p^{j0}_{0},\dots p^{j0}_{r})\dots;\alpha(p^{jq}_{0},\dots p^{jq}_{s}),$$

$$;\alpha(p^{j0}_{0},\dots p^{j0}_{r}) = ;\alpha(p^{j0}_{0}),\dots \alpha(p^{j0}_{r}),$$

If activation resolves as amplitudes, then

$$\alpha(p^{ji}_{k(i)}) \rightarrow |p^{ji}_{k(i)}| \in R.$$

For convenience we can normalize the activation scalar residue such that,

$$\underline{\alpha}(p^{ji}_{k(i)}) \equiv \alpha(p^{ji}_{k(i)})/|p^{ji}_{k(i)}| \rightarrow 1.$$

Using this definition of activation, let us now include the entire process of activation with memorization. To incorporate memorization designate module *activation* as

$$\alpha(\varphi_i[\mathbf{c}^i] \Rightarrow \boldsymbol{c}^i \Rightarrow \phi^i[\eta\boldsymbol{c}^i]) = \alpha(\varphi_i[\mathbf{c}^i]) \Rightarrow \alpha(\boldsymbol{c}^i) \Rightarrow \alpha(\varphi^i[\eta\boldsymbol{c}^i])$$

$$\eta\underset{\cdot}{c}^i \rightarrow \alpha(\boldsymbol{c}^i) \rightarrow \eta\underset{\cdot}{c}^i \Rightarrow \mathbf{c}^i.$$

Write this as a simple juxtaposition of parse-activate-concatenate:

$$\varphi_i[\mathbf{c}^i](\alpha(\boldsymbol{c}^i) \rightarrow \eta\boldsymbol{c}^i)\varphi^i[\eta\boldsymbol{c}^i] = \eta\boldsymbol{c}^i(\alpha(\boldsymbol{c}^i) \rightarrow \eta\boldsymbol{c}^i)\mathbf{c}^i.$$

The 1st term of the *sequence moments* is the **c**–string decoding as a cluster replica; the 2nd term is the 'activation' of the control cluster, which is the passing of commands to the system hardware and the cluster replica; the 3rd term is the replica cluster coding as **c**–string: this term is the reconstruction of the (perhaps novel) **c**–string. This designates the total action of the activation-memorization cycle. The total sequence through all modules for field generation has the form:

$$S^{ji}(\alpha(\varphi_i[\mathbf{c}^{ji}](\alpha(\boldsymbol{c}^{ji})\rightarrow\eta\boldsymbol{c}^{ji})\varphi^i[\eta\boldsymbol{c}^{ji}])) = S^{ji}(\alpha(\eta\boldsymbol{c}^{ji}(\alpha(\boldsymbol{c}^{ji})\rightarrow\eta\boldsymbol{c}^{ji})\mathbf{c}^i.$$

We assume sequencing is unitary, i.e.,

$$S^{ji}(\eta\boldsymbol{c}^{ji}\,\alpha(\boldsymbol{c}^i)\,\mathbf{c}^i) = \eta\boldsymbol{c}^{ji}\,\alpha(\boldsymbol{c}^{ji})\,\mathbf{c}^{ji}.$$

If we assume recollection and restore can happen 'continuously', this shows a simultaneous decoding of **c**–strings as cluster replicas and the coding of **c**–strings as replicas: a simultaneous construction and deconstruction of memory. This decode-encode cycle is the core of memorization-activation. There are variations on the 'path' of this cycle which we will discuss below.

Lattice Algebra

For elements a, b & c of the set of strings, $\{\sigma\}$, with cardinality $|\{\sigma\}|$ we have a partially ordered set, or poset, relation:

$$a\Re a\in|\{\sigma_\kappa\}|;$$

$$a\Re b = b\Re a\in|\{\sigma_\kappa\}|\text{: } a=b;$$

$$a\Re b \ \&\ b\Re c\text{: } a\Re c\in|\{\sigma_\kappa\}|;\ |\{\sigma_\kappa\}|,\ \kappa\in N.$$

$|\{\sigma_\kappa\}|$ designates the cardinality of κth subset. The relation, $\Re$, is 'greater then or equal' ($\geq$) & it is constructed via composition which is 'realized' in the filter group action. If we designate the string 'magnitude' as the set cardinality we have

$$|\{\sigma_\kappa\}|\Leftrightarrow|\Phi_{\kappa*}|.$$

The cardinality of the set, $|\{\sigma\}|$ has an a 'natural order', or ordinality formed by the cardinality of its poset subsets, $|\{\sigma_\kappa\}|$: the order of cardinality on the poset is total. If we define a magnitude as the subset cardinality we have an ordered measure on $\{\sigma\}$ called a *lattice*. We may designate this lattice as

$$L(|\{\sigma\}|,\geq)\Leftrightarrow L(|\Phi|,\geq).$$

By duality,

$$L(|\{\sigma\}|,\leq)\Leftrightarrow L(|\Phi^{-1}|,\geq),$$

also forms a poset. There is also a lattice of relation, $\Re$ is 'string set inclusion' ($\subseteq$). Actually our sets are such that we have proper inclusion ($\subset$) & the magnitudes satisfy the non-equality relations: ($>,<$). The memory lattice, $L(\{\sigma\},\subset)$, is realized as the filter group action as this set relation is generated by the filter action we may designate this as,

$$L(\{\sigma\},\subset)\Leftrightarrow L(\{\sigma\},\Phi).$$

Again, by duality:

$$L(\{\sigma\},\not\subset)\Leftrightarrow L(\{\sigma\},\Phi^{-1}).$$

The memory filters generate the lattice. The 'exclusion' ($\not\subset$) has a specific relation between elements that satisfies conjunction-disjunction logical operators, i.e., the exclusive sets generated satisfy an 'set orthogonality' condition:

$$\cup_{i\neq j}\,\phi^{-1}{}_j\cap\phi^{-1}{}_i=\varnothing.$$

We also know that

$$\cup_{i\neq j}\,\phi^{-1}{}_j\cap\phi_i=h^-,$$

where h is a 'residue' of filtration. If we designate this a 'backward residue', h^-, we also have a forward residue:

$$\cup_{i\neq j}\,\phi_j\cap\phi^{-1}{}_i=h^+.$$

These residuum are not elements of the control sequence sets, i.e., they are not representations, but merely expressions of the filter syntax operating on arbitrary strings. If we include a difference operator we have,

$$\cup_{i\neq j} (\phi_j \cap \phi^{-1}{}_i - \phi^{-1}{}_j \cap \phi_i) = h^+ - h^- = \delta h,$$

where the difference of sets: $\{a\} - \{b\}$ is the set $\{x \in a: x \notin b\}$. The set δh is a residue of filtration as an error set. We can designate this operation as a commutator of filters:

$$[\phi_j, \phi^{-1}{}_i] = \delta h.$$

The anti-commutator is a set $\{a,b\} \equiv \{a\} + \{b\}$, which we can assign as set union. We thus have an exclusion-inclusion operator group:

$$[\phi_j, \phi^{-1}{}_i] = \delta h.$$

$$\{\phi_j, \phi^{-1}{}_i\} = \Delta h.$$

These residue sets form 'novel' strings of data which may or may not form representations for a control sequence. These 'theorem generators' will in general form 'noisy data'. This Lie algebra is reminiscent of Heisenberg's algebra with Planck's constant serving as residue of 'operations'; an interesting aside is that the original control algebra was designed to create Doppler fields(!). Suppose the strings designated residual patterns of a transduction-filtration: could these novel patterns of filtration 'eventually' represent (map to) something? Yes, if we build a system to 'use' them in some 'meaningful way'. Syntactically we can build an acceptor that accepts all residuals; semantically they may signify nothing. What we have with residues is filters operating on 'unintended' strings...perhaps 'associating' data & a filter(?).

Do we have an algebraic lattice: $L(\{\sigma\},\Phi,\Phi^{-1})$? What about the 'extended' lattice: $L(\{\sigma\},[\Phi,\Phi^{-1}],\{\Phi,\Phi^{-1}\})$, or even the more 'complex': $L(\{\sigma\},\Phi,\Phi^{-1},[\Phi,\Phi^{-1}],\{\Phi,\Phi^{-1}\})$? What information can be embedded/extracted from these lattice algebras?

*

For a Boolean algebra we require two binary operators ($\cap,\cup$) & four laws:

1. commutativity
2. associativity
3. absorption
4. indempotency

We can exclude commutativity & associativity & still create algebras. If we define union & intersection as supremum & infinum respectively then we have an algebraic lattice; we can do with cardinality, or the magnitude of the sets; but what about the 'amplitudes', i.e., the strings themselves? Can the filters form a unique algebra?

Concatenation can 'stand' for set union, or 'vector sum'; but parsing is not so simply assumed as set intersection; but it does reduce the larger set, i.e., the vector to its 'basis'. So, can w say the intersection of elements of a set are the bases? Can we map intersection into projection?

The lattice also has an *energy* measure & order under the partial relation. If assign dimensions on the module & its clusters we can also define an inner-product. For example, a cluster measure on its parameter set is,

$$E(p^i_{k(i)}) \rightarrow \Sigma_{ik}|p^i_{k(i)}|^2 \in R.$$

We have a poset of energy on this lattice. We may define this as an inner-product norm on the cluster, thus equating a cluster with a 'data vector'; therefore, the energy measure of a cluster is also defined as

$$E_c \equiv \mathbf{c}^i \bullet \mathbf{c}^i = \Sigma_{ik} p^i_{k(i)}{}^2.$$

Analogously, for the filters,

$$E_m \equiv \cup_\kappa \phi_\kappa \bullet \phi^{\kappa-1}.$$

Where ϕ_κ generates string σ_κ from 'below', i.e., control cluster strings ($\mathbf{c}^i$); while $\Phi^{\kappa-1}$ generates the same sting from 'above', i.e., the T-Seq memory string ($\mathbf{M}^n{}_{jik}$) (cf. below): the filters operate between memory co-domains. The module basis would be assigned analogously to control clusters, with bases for generator, modulator & switch. If we partition the set union, $\cup_\kappa \phi_\kappa \bullet \phi^{\kappa-1}$, over a basis of subsets, we have the module energy 'spectrum' designated as,

$$E_m \equiv \gamma_\kappa \bullet \gamma^{\kappa-1} + \mu_\kappa \bullet \mu^{\kappa-1} + \xi_\kappa \bullet \xi^{\kappa-1}.$$

Here we have a generator (γ), modulator (μ) & switch (ξ) energy respectively. The total 'memory energy', E_M is the sum of all module energies,

$$E_M \equiv \Sigma E_m.$$

This energy, of course designates 'informational energy', not 'material energy'.

The basis vectors for the clusters are the parameter types; for a modulator & RF field generators these parameters are amplitude, carrier frequency, modulation sets (frequency, deviation, etc.) & 'other' logical controls. We are free to define measures on logic as maps of the logical constants into the set $\{0,1\} \in N \in R$. All frequencies may share a common basis, amplitude is 'orthogonal' to frequency, i.e., we define it thus on the clusters. Note that there is phasing of the frequency-time cells, but the control structure for a field generator coordinates phasing thru output port gating; which I'll not discuss here, as it is not relevant to the control structure.

The energy of the clusters, as elements of a 'module energy forms a lattice, i.e., we have a *minimal* & *maximal* energy measure within the sets, i.e., within & among clusters, within the modules & among modules & within the global configuration of modules. There is a greatest element: the global string cardinality, $|\mathbf{M}^{n}_{jik}|$. There is no smallest element, as the minimal elements are discrete control clusters.

If we include the null string, $\mathbf{s}^{\varnothing}$, in the set of all strings resolved by the filters of this system, these strings form a group without identity under concatenation, i.e., there are no inverse strings under the rule of concatenation. We shall see below that the string filters form a two element group with identity as string operators between co-domains & the channel of filters forms a group lattice.

Memorization Filter Action

We may designate the module activity as a sequence: *recall-activate-learn*; but, we may also look at recall and learning as 'simultaneous' events, i.e., we can bypass activation; the designation is simply: $\varphi_i \circ \varphi^i[\eta \boldsymbol{c}^i]$. Note that these are the two moments of a construction-deconstruction sequence, i.e., $\eta \boldsymbol{c}^i$ and $\mathbf{c}^i$; with composition this resolves as,

$$\varphi_i \circ \varphi^i[\eta \boldsymbol{c}^i] \Rightarrow \eta \mathbf{c}^i.$$

The 'reverse' action is,

$$\varphi^i \circ \varphi_i[\mathbf{c}^i] \Rightarrow \mathbf{c}^i.$$

This is a group action which we'll discuss below.

A sequence of parameters forms a cluster activation; a sequence of cluster activations form the module activation; as a sequence of modules activations forms the total test sequence: S(**M**). With the above approach of activation-memorization, the module action is,

$$S^{ji}(\eta \boldsymbol{c}^{ji}(\underline{\alpha}(\boldsymbol{c}^{ji}) \rightarrow \eta \boldsymbol{c}^{ji})\mathbf{c}^{ji}) \Rightarrow \alpha(\vec{\vec{\mathbf{m}}}^{j}).$$

This designates the activation of all clusters as 'carried' by memory module $\vec{\vec{m}}^{j}$. Since,

$$\cup^{i}\mathbf{c}^{ji} \equiv \vec{\vec{\mathbf{m}}}^{j},$$

Thus the 'action' has a residual memory:

$$\underline{\alpha}(\overleftrightarrow{m}^{j}) \Rightarrow \overleftrightarrow{\mathbf{m}}^{j}.$$

The resultant is the module string memory of cluster strings, i.e., the module memories. The 'set' of all **m**–module activation-memorization is:

$$\underline{\alpha}(\varphi_i\phi_j\gamma_{\mathbf{M}}[\mathbf{M}]\gamma^{\mathbf{M}}\phi^{j}\varphi^{i}[\eta \boldsymbol{c}^{i}])$$

$$= \underline{\alpha}(\varphi_i\phi_j[\overleftrightarrow{\mathbf{m}}^{j}]\gamma^{\mathbf{M}}\phi^{j}[\mathbf{c}^{ji}])$$

$$= \underline{\alpha}(\varphi_i[\mathbf{c}^{i}]\gamma^{\mathbf{M}}[\overleftrightarrow{\mathbf{m}}^{j}]) = \underline{\alpha}(p^{ji}{}_{k(i)}\mathbf{M}^{ji}{}_{k(i)})$$

$$= \underline{\alpha}(p^{ji}{}_{k(i)}\mathbf{M}^{ji}{}_{k(i)}) = \underline{\alpha}(p^{ji}{}_{k(i)})\mathbf{M}^{ji}{}_{k(i)} \Rightarrow \mathbf{M}^{ji}{}_{k(i)}.$$

This is the global string memory, as expected. As discussed above, the activation is not simply the total memory; but rather a sequence through this set union. We may designate the test sequence,

$$S(\mathbf{M}^{ji}{}_{k(i)}) \Leftrightarrow S^{j}(S^{i}(S_{k(i)}(\underline{\alpha}(p^{ji}{}_{k(i)})))) \equiv S^{ji}{}_{k(i)}(p^{ji}{}_{k}).$$

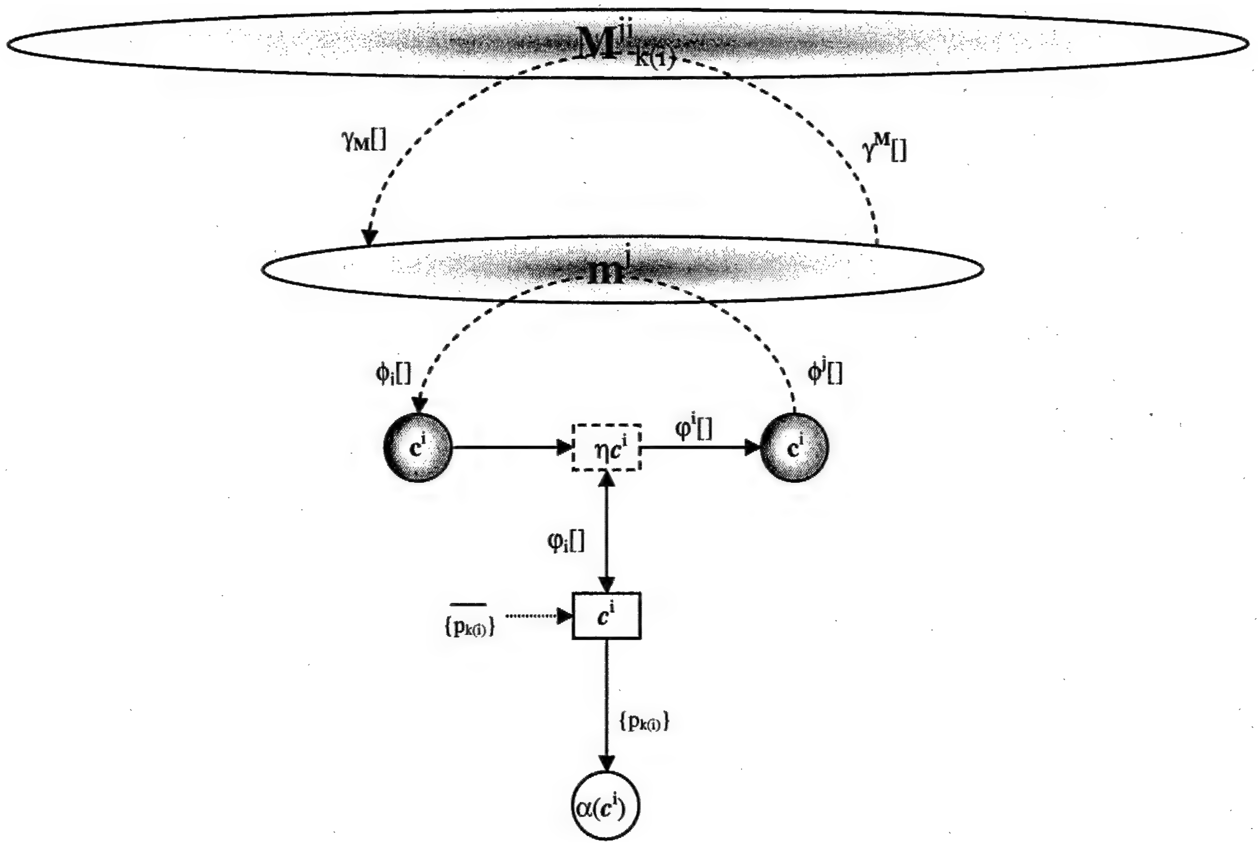
Activation has the *effect* of activating the RF equipment and re-memorizing the control structure string. Under this structure, each memorization on parameter change requires a re-concatenation (and re-store) and (re-open) re-parse; this is not the original structure; but a new approach used in this particular investigation. In simple terms we have learn and recollection filters active at all times. The control cluster is the 'point of control' for both the operator and system restoration: the system state is memorized from the control cluster structure as the parameter set and returned there for activation. There are two 'access paths' into the control structure – the replica cluster recollected from memory and the operator interface. We denote the operator input as the 'lift' of parameters into clusters; this *lift* is designated:

$$\overline{\{p^{ji}_{k(i)}\}} \underline{\Rightarrow} \boldsymbol{c}^{i}(\{p_{k(i)}\}).$$

The data initially gets into the system via the operator; all subsequent changes also come from the operator. The memory architecture only recalls and restores.

Activation and Memorization

The **m**–modules are equivalent to the **m**–strings; the **c**–clusters are equivalent to the **c**–strings, and parameters to p–strings; the activation of a sequence, S(**M**) is equivalent to the activation of its modules in some set sequence: S(μ) of the module cluster parameter measure sets. A picture of the architecture is shown here.



If activation occurs with the distribution of parameters, a sequence is equivalent to a memory deconstruction; if activation also restores memory in an ascending chain, then activation is also equivalent to a memory construction. If both conditions apply, i.e., activation occurs on memorization, then activation is equivalent to memorization.

We defined the core, or 'inner-action' as,

$$S^{ji}(\eta c^{ji}(\underline{\alpha}(c^{ji})\rightarrow\eta c^{ji})\mathbf{c}^{ji}).$$

This implies an 'outer-action'. Note that outer-filters construct and deconstruct the global **M**–string; also note the lift of p–strings, while not a filter action may also designate an outer-action, i.e., the extremes of the memory hierarchy. The 'intermediate' filters construct and deconstruct **m**–strings. For this architecture, we have two inner actions

$$\underline{\alpha}(\phi_i \circ \phi^i[\eta c^i]) \Rightarrow \mathbf{c}^i,$$

and

$$\underline{\alpha}(\phi^i \circ \phi_i[\mathbf{c}^i]) \Rightarrow \eta c^i,$$

These *moments* are the two results of filtering which depend on the 'direction' of filtration. If the extender operates first, the result is the constructed string; otherwise, the result is the cluster replica. Each side of the filter operates in opposite direction, yielding the same memory in 'conjugate forms'. If we allowed for simultaneous extension and contraction with some 'persistence', the result is

$$\underline{\alpha}(\phi_i[\mathbf{c}^i]\phi^i[\eta c^i]) \Rightarrow \eta c^i\mathbf{c}^i$$

Thus the total 'forward' and 'reverse' action is

$$\underline{\alpha}(\phi_i[\mathbf{c}^i]\phi^i[\eta c^i]) + \underline{\alpha}(\phi^i[\eta c^i]\phi_i[\mathbf{c}^i]) \Rightarrow \eta c^i\mathbf{c}^i + \mathbf{c}^i\eta c^i.$$

If these were to form a group, the difference is

$$\underline{\alpha}(\phi_i[\mathbf{c}^i]\phi^i[\eta c^i]) - \underline{\alpha}(\phi^i[\eta c^i]\phi_i[\mathbf{c}^i]) \Rightarrow \eta c^i\mathbf{c}^i - \mathbf{c}^i\eta c^i.$$

There is an algebra waiting to be defined for these elements; tho I won't go into that here. It is interesting that these look like exchange equations. ☺. There are two outer filter actions which yield respectively,

$$\underline{\alpha}(\gamma_{\mathbf{M}}[\mathbf{M}]\phi^{j}[\dddot{\mathbf{m}}^{j}]) \Rightarrow \dddot{\mathbf{m}}^{j},$$

$$\underline{\alpha}(\phi^{j}[\dddot{\mathbf{m}}^{j}]\gamma_{\mathbf{M}}[\mathbf{M}]) \Rightarrow \mathbf{M}.$$

Again, if we allowed for simultaneous action with persistence the result is

$$\underline{\alpha}(\gamma_{\mathbf{M}}[\mathbf{M}]\phi^{j}[\dddot{\mathbf{m}}^{j}]) \Rightarrow \dddot{\mathbf{m}}^{j}\mathbf{M},$$

$$\underline{\alpha}(\phi^{j}[\dddot{\mathbf{m}}^{j}]\gamma^{-\mathbf{M}}[\mathbf{M}]) \Rightarrow \mathbf{M}\dddot{\mathbf{m}}^{j}.$$

The direction of operation for sequential action yields one or the other forms of memory. There are other actions, e.g.,

$$\underline{\alpha}(\varphi^{i}[\mathbf{c}^{i}]\gamma^{\mathbf{M}}[\dddot{\mathbf{m}}^{j}]) \Rightarrow \eta \boldsymbol{c}^{i}\mathbf{M}^{ji}{}_{k(i)}.$$

We could define the entire algebra. Note that we have ascending and descending sequences:

$$\mathrm{p}^{ji}{}_{k(i)} \rightarrow \eta \boldsymbol{c}^{i} \Rightarrow \mathbf{c}^{i} \Rightarrow \dddot{\mathbf{m}}^{j} \Rightarrow \mathbf{M}^{ji}{}_{k(i)};$$

$$\mathrm{p}^{ji}{}_{k(i)} \leftarrow \eta \boldsymbol{c}^{i} \Leftarrow \mathbf{c}^{i} \Leftarrow \dddot{\mathbf{m}}^{j} \Leftarrow \mathbf{M}^{ji}{}_{k(i)}.$$

This shows all successive moments 0th to 3rd order. If we take the union of the parameter sets we have the global memory,

$$\cup^{ji}\, \mathrm{p}^{ji}{}_{k(i)} \equiv \mathbf{M}^{ji}{}_{k(i)}.$$

This is the total memory in the form of parameter sets and as the global string; sequences 'distribute' parameters to there respective module-clusters. This 'in and out' of memory parameters is accomplished by the opposing filters yield a quadrature of memory forms. If we align the filters with the moments we have,

$$\begin{tikzcd}
p^{ji}_{k(i)} \arrow[r] & \eta \boldsymbol{c}^i \arrow[r, Rightarrow] & \mathbf{c}^i \arrow[r, Rightarrow] & \overleftrightarrow{\mathbf{m}}^j \arrow[r, Rightarrow] & \mathbf{M}^{ji}_{k(i)}; \\
 & & \varphi^i \arrow[u] & \phi^j \arrow[u] & \gamma^{\mathbf{M}} \arrow[u] \\
 & & \varphi_i \arrow[u, leftrightarrow] \arrow[d] & \phi_j \arrow[u, leftrightarrow] \arrow[d] & \gamma_{\mathbf{M}} \arrow[u, leftrightarrow] \arrow[d] \\
p^{ji}_{k(i)} & \eta \boldsymbol{c}^i \arrow[l] & \mathbf{c}^i \arrow[l, Rightarrow] & \overleftrightarrow{\mathbf{m}}^j \arrow[l, Rightarrow] & \mathbf{M}^{ji}_{k(i)}. \arrow[l, Rightarrow]
\end{tikzcd}$$

Quadrature Group Filters

In RTEx S(**M**) these moments were designed as mutually exclusive, i.e., there is one and only one moment: a recollect or restore active in any sequence. This serial action is an construction-deconstruction of memories under filter action and has a general form: $\psi_s\psi^s$; As each filter element has an inverse and the composition of inverses on an element reproduces the element, the opposing action resolves as a unit: $\psi_s\psi^s \Leftrightarrow 1$ & $\psi_s\psi^t \Leftrightarrow 0$: thus the naming convention[1].

$$\varphi_i \circ \varphi^i[p^{ji}_{k(i)}] \Rightarrow p^{ji}_{k(i)},$$

$$\phi_j \circ \phi^j[\mathbf{c}^i] \Rightarrow \mathbf{c}^i,$$

$$\gamma_{\mathbf{M}} \circ \gamma^{\mathbf{M}}[\overleftrightarrow{\mathbf{m}}^j] \Rightarrow \overleftrightarrow{\mathbf{m}}^j.$$

The 1st group is the inner-group, the latter two are the outer-groups. As the filter action is symmetric we have a 'left composition' which is a composition in the 'reverse direction'; in this case:

$$\varphi^i \circ \varphi_i[\mathbf{c}^i] \Rightarrow \mathbf{c}^i.$$

[1] The name is derivative of orthogonal signal filters, tho these data filters are not a perfect analogy.

This is the inner-group. For left composition the outer-groups are:

$$\phi^j \circ \phi_j[\overleftrightarrow{\mathbf{m}}^j] \Rightarrow \overleftrightarrow{\mathbf{m}}^j,$$

$$\gamma^{\mathbf{M}} \circ \gamma_{\mathbf{M}}[\mathbf{M}^{ji}{}_{k(i)}] \Rightarrow \mathbf{M}^{ji}{}_{k(i)}.$$

The total group action designates the action of memorization as it represents the collection, recollection and restoration. The group action is non-commutative and the inverse groups meet in the cluster, thus the designation *center of action* for the forward-backward quadrature set, $\varphi_i\varphi^i$. Even if we remove the replica and define a filter that lifts parameter sets directly into cluster strings, we have the forward group:

$$\varphi_i \circ \varphi^i[p^{ji}{}_{k(i)}] \Rightarrow p^{ji}{}_{k(i)},$$

$$\phi_j \circ \phi^j[\mathbf{c}^i] \Rightarrow \mathbf{c}^i,$$

$$\gamma_{\mathbf{M}} \circ \gamma^{\mathbf{M}}[\overleftrightarrow{\mathbf{m}}^j] \Rightarrow \overleftrightarrow{\mathbf{m}}^j.$$

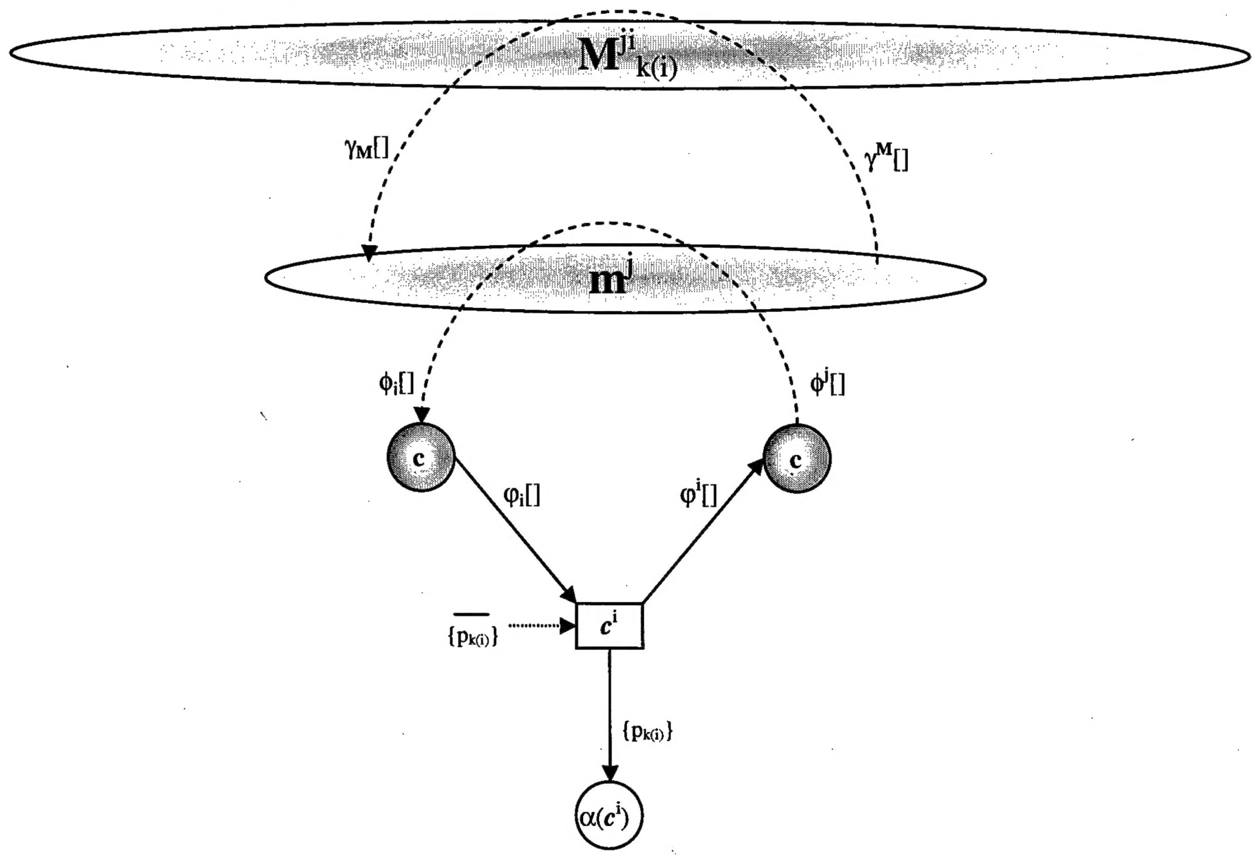
And the backward groups are,

$$\varphi^i \circ \varphi_i[\mathbf{c}^i] \Rightarrow \mathbf{c}^i,$$

$$\phi^j \circ \phi_j[\overleftrightarrow{\mathbf{m}}^j] \Rightarrow \overleftrightarrow{\mathbf{m}}^j,$$

$$\gamma^{\mathbf{M}} \circ \gamma_{\mathbf{M}}[\mathbf{M}^{ji}{}_{k(i)}] \Rightarrow \mathbf{M}^{ji}{}_{k(i)}.$$

Thus, $\psi_i \circ \psi^i$ remains the center of action for all quadrature filter pairs. In this case the memory architecture is as shown below.



The filters form a group of sub-groups over the parameter set, $p^{ji}{}_{k(i)}$, and memory set, **M**. We designate the entire forward group filter as $\Phi[\,]$ and the backward group as $\Phi^{-1}[\,]$. In the forward direction, the action begins with the parameter sets to yield the global string, i.e., as

$$\Phi[p^{ji}{}_{k(i)}] \Rightarrow \mathbf{M}^{ji}{}_{k(i)}$$

In the backward direction the action is,

$$p^{ji}{}_{k(i)} \Leftarrow \Phi^{-1}[\mathbf{M}^{ji}{}_{k(i)}]$$

Also, as $\cup^{ji}{}_{k(i)}\ p^{ji}{}_k \equiv \mathbf{M}^{ji}{}_k$ has a *supremum*, $\mathbf{M}^{ji}{}_{k(i)}$ and *infinum*, $\varnothing$, the memory forms a lattice; in fact, each subgroup is a lattice all with null *infinum* thus all intersect in the null. The above action designates a 'right composition'. This is the action for sequential, or serial activation;

what if we look at the successive moments as parallel? The algebraic structure over the lattice will define the meaning of the memory.

Summary and Suggestions

From all this we can see that the memory structure I've built has memory structure, as a 3-dimensional object of control and memorization. The S(**M**) is a composite of modules, which are composites of control clusters, which are composites of parameter sets. We have an ordered grade of memory objects which form not only a lattice, but also a group hierarchy of subgroups. By mapping a global string, $\mathbf{M}^{ji}{}_{k(i)}$, through a sequence of data filters we have also shown the equivalence of a memory global string file, $f(\mathbf{M}^{ji}{}_{k(i)})$, and a sequence, $S(\mathbf{M}^{ji}{}_{k(i)})$, through its contents. We may designate this equivalence as

$$f(\mathbf{M}^{ji}{}_{k(i)}) \approx \Phi \circ \Phi^{-1}[\mathbf{M}^{ji}{}_{k(i)}] \approx \Phi^{-1} \circ \Phi[p^{ji}{}_{k(i)}] \approx S(\mathbf{M}^{ji}{}_{k(i)}).$$

The purpose was to investigate the nature of memory as simultaneously providing memory structural form while modifying existing memory content, which is more closely aligned with organic (biological) memory rather then mechanistic (logical) memory. In fact, my model was the DNA 'control center' and RNA 'filters'. In my approach, all the configuration information is contained in a global string, while data filters open the string and distribute the information to modules. In this light, a module is analogous to a ribosome, which processes the control string with respect to a command set (amino acid) to generate Doppler fields (protein sequences). Of course, the analogy ends here.

We can perhaps extend this technique to include 'simultaneous learning-action'. If we assume memory modules updating anytime a parameter changes in any of its clusters, and assume quadrature filters operate simultaneously over global memory and parameter sets, we have a dynamical system, providing parameters on recall and remembering changes as they occur. The method of learning via quadrature data filters requires further investigation into applicable algebras, where the goal is activity as quadrature filters with *residual of action* that is novel memory. I have incorporated this technique with a supplementary information channel the codes an 'external environment; the supplementary channels allow for an associate memory structure that links control sets to environmental parameters for adaptive reconfigurations.

References:

Basic Algebra I, Jacobson, Nathon; W.H. Freeman and Company, New York, 1985.

[i] Cf. Chomsky's language hierarchy; 3-level languages are equivalent to automata.
[ii] RF Test Executive.
[iii] ':', ';', ',' all designate string element delimiters.